REMARKS

The Final Office Action, mailed August 15, 2006, considered and rejected claims 1-4, 7, 10-17 and 20-28. Claims 1, 2, 4, and 7-27 were rejected under 35 U.S.C. 102(b) as being anticipated by the TETware Release 3.3 software product released September 18, 1998, by The Open Group, as evidenced by: "TETware User Guide, Revision 1.2", "Release Notes for TETware Release 3.3" and "TETware Programmers Guide, Revision 1.2". Claim 3 was rejected under 35 U.S.C. 103(a) as being unpatentable over TETware and the associated cited documentation as applied to claim 1 above, and further in view of Hartmann et al. (U.S. Patent No. 6,505,342).[1]

By this amendment, claims 1, 4, 7, 10-12, 15, 20, 21 and 24 have been amended and claim 28 has been cancelled. Accordingly, claims 1-4, 10-17 and 20-27 are pending, of which claims 1, 4, 15 and 24 are the only independent claims at issue.

Applicants' invention, as recited for example in independent system claim 1, relates to a computer system for selecting and organizing individual test cases for use in testing a computer program to ensure that the program processes as intended. The system includes one or more program modules storing a plurality of available test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface; a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) initiates execution of a connector to scan for and discover the plurality of available test cases that are stored in the one or more program modules and to organize the plurality of available test cases into a test case hierarchy, and (iii) receives user input indicating which of the plurality of available test cases in the test case hierarchy are selected test cases to be executed on the computer program; a harness comprising a set of instructions that (i) receives the test case hierarchy including the one or more program modules storing the plurality of test cases, (ii)

[1] Although the prior art status and some of the assertions made with regard to the cited art is not being challenged at this time, inasmuch as it is not necessary following the amendments and remarks made herein, which distinguish the claims from the art of record, Applicants reserve the right to challenge the prior art status and assertions made with regard to the cited art, as well as any official notice, which was taken in the last office action, at any appropriate time in the future, should the need arise, such as, for example in a subsequent amendment or during prosecution of a related application. Accordingly, Applicants' decision not to respond to any particular assertions or rejections in this paper should not be construed as Applicant acquiescing to said assertions or rejections.

traverses the test case hierarchy, and (iii) executes each of the selected test cases using the corresponding language and format independent interface of the selected test case to ensure that the computer program processes as intended; the connector, initiatable by the harness client, and comprising a set of instructions that (i) scans for the plurality of available test cases stored in the one or more program modules, (ii) organizes the plurality of available test cases into the test case hierarchy by extracting the plurality of available test cases from the one or more program modules, and (iii) selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written; and a processor for executing each selected test case, the harness, the harness client, and the connector, such that a first test case written in a first language and a second test case written in a second, different language are each executable by the processor because of the language and format independent interface.

Applicants' invention, as recited for example in independent method claim 4, relates to testing a computer program to determine whether the computer program processes as intended. The method includes a harness client (i) receiving user input that specifies one or more filenames to identify the program module, (ii) initiates execution of the connector to scan for and discover the plurality of test cases of interest that are stored in the program module and to organize the plurality of test cases of interest into a test case hierarchy, and (iii) receives user input indicating that at least two of the plurality of test cases of interest in the test case hierarchy are the selected test cases to be executed on the computer program; a connector scanning the plurality of test cases of interest stored in the program module, each test case having a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case, wherein the connector is initiatable by the harness client; the connector extracting the plurality of test cases of interest from the program module; the connector organizing plurality of test cases of interest into the test case hierarchy; the connector interfacing the harness with the selected test cases, wherein the interfacing allows the harness to recognize and execute the selected test cases regardless of the language or format in which the plurality of test cases of interest was developed; and a harness traversing the test case hierarchy and executing each of the selected test cases to test the computer program, such that a first test case written in a first language and a second test case written in a second, different language are each executable by the harness because of the language and format independent

interface. Independent claim 15 recites similar limitations from the perspective of a computer program product.

Applicants' invention, as recited for example in independent method claim 24, similarly relates to testing a computer program to determine whether the computer program processes as intended. The method includes specifying one or more filenames for identifying one or more program modules storing a plurality of test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface; identifying the plurality of test cases within the one or more program modules; translating the identified plurality of test cases into a test case hierarchy that includes the plurality of available test cases, a first test suite, a second test suite, and a test module, the first test suite including at least two of the plurality of the available test cases, the second test suite including at least another one of the plurality of the available test cases, and the test module including the first and second test suites; indicating that the plurality of test cases in the test case hierarchy is to be executed on the computer program; providing an interface to the test case hierarchy in order to recognize and execute the plurality of test cases regardless of the language or format in which the plurality of test cases was written; and running each of the plurality of test cases in the test case hierarchy to test the computer program, wherein a first test case written in a first language and a second test case written in a second, different language are each executable on the computer program because of the language and format independent interface.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." MPEP § 2131. That is, "for anticipation under 35 U.S.C. 102, the reference must teach every aspect of the claimed invention either explicitly or impliedly." MPEP § 706.02. Applicants also note that "[i]n determining that quantum of prior art disclosure which is necessary to declare an applicant's invention 'not novel' or 'anticipated' within section 102, the stated test is whether a reference contains an 'enabling disclosure.'" MPEP § 2121.01. In other words, a cited reference must be enabled with respect to each claim limitation. During examination, the pending claims are given their broadest reasonable interpretation, *i.e.*, they are interpreted as broadly as their terms reasonably allow, consistent with the specification. MPEP §§ 2111 & 2111.01.

As noted in Applicants' prior response, TETware discloses grouping test cases within a test suite. TETware PG, section 2.2. Test suites are organized as directory hierarchies—the top

of each test suite directory is known as the test suite root directory. TETware UG, section 5.2.6. All files in a test suite reside below the test suite directory (or in a specified alternate execution directory). TETware PG, section 2.3; TETware UG, section 5.2.6. Test suites are required to include certain files and utilities, such as a build tool (e.g., makc), a clean tool (e.g., rm), at least one test scenario file, etc. TETware PG, section 2.5.

A test scenario is a list of invocable components from a test suite that are processed during a particular TETware invocation. TETware UG, sections 2.2 & 5.3.2. Within a scenario file, a test case name may appear by itself or be attached to a directive that describes how the test case should be executed (sequentially, in parallel with other test cases, repetitively, remotely, etc.). TETware PG, section 4.2.4.3; TETware UG, section 5.3.2.2. Test case names are interpreted relative to the test suite root directory or alternate execution directory, depending on the mode of operation. TETware UG, section 5.3.2.4. Section 5.3.2.5 of the TETware UG and section 4.4 of the TETware PG present some simple examples of test scenarios. Example test cases names listed in these scenarios follow the directory convention explained above (e.g. "/tset/tc1" and "/ts/tc1").

The test cases in a test suite are processed by a Test Case Controller (TCC), based on a chosen mode of operation (e.g., build, execute, clean up). TETware PG, section 2.5. In build mode, the TCC translates source test cases into executables, in execute mode the TCC loads and executes test cases, and in clean mode the TCC removes unwanted files. TETware PG, section 3.2. Each test case is an executable program. TETware PG, section 2.2. When a test case uses one of the TETware language specific APIs, its execution is supervised by a Test Case Manager (TCM). TETware PG, section 2.4.2. The TCM is not a separate program, but instead is linked with user-supplied test code and the API library to produce an executable test case. *Id.* There is a separate TCM module for each API that is supported by TETware. *Id.* For example, TETware includes a C TCM and a C++ TCM. TETware PG, section 2.4. Test cases are written to a specific language binding (C, C++, Shell, Korn Shell, etc.). TETware PG, sections 8, 9, 10, and 11.

The Decision on Appeal, mailed August 29, 2008, stated the following in regard to the teachings of TETware:

> As set forth in the Findings of Fact section, TETware discloses a tcc that builds a designated test case when it operates in the build mode. (FF. 5.) Further, TETware

> discloses a test case hierarchy having a root node directory, which includes one or more test case suites, each of which including in turn one or more test cases. (FF.2.) Additionally, TETware discloses that in the execute mode, the tcc executes designated test cases in the test case hierarchy. (FF. 5-6.) In the above limited situation, we find that when the tcc is operating in the build mode, it creates at least one designated test case. Therefore, tcc creates a hierarchy of one test case in the build mode. Further, in the above limited situation, in the build mode, tcc generates the hierarchy of a plurality of test cases that may be clustered as test cases suites, which in turn may be clustered as a test case module. Therefore, TETware's disclosure teaches scanning a program module to create a hierarchy of a single test. In other words, TETware teaches scanning a program module for a single test arranged as a hierarchy of one. Decision on Appeal, pp. 10-11.

The Appeal Board's repeated use of the phrase "in the above limited situation" indicates that the reasoning presented applies only in the limited situation where a program module is scanned for a single test arranged as a hierarchy of one. As further indicated in the Decision, "In a nutshell, we construe the cited claimed limitations to broadly but reasonably read on scanning the program module for a single test case arranged as a hierarchy of one." Decision on Appeal, p. 10. Accordingly, the Decision, in its broad reading of the claim language, carved out a single scenario in which TETware could reasonably be construed to read on the pending independent claims.

As a result of the identified "limited situation" where a program module is scanned for a single test case arranged as a hierarchy of one, Applicant's have amended the claim language of the independent claims to no longer include a situation where a "hierarchy of one" is possible. Claims 1, as well as the other independent claims, now clearly recites a test hierarchy in which one or more program modules store a plurality of available test cases. Thus, as currently amended, the independent claims cannot include the limited situation identified by the Appeal Board, namely, a hierarchy of one.

Moreover, in order to further distinguish the test module of the present invention from TETware's directory element, $TET_ROOT, the phrase "the test module being executable by a harness" has been added to the independent claim 24. (Support for this limitation can be found at least on page 17, ll. 1-7 of the specification). As indicated in the specification, the test cases, the

test suite and the test module are each separately and individually capable of being executed by a harness. TETware's directory element, $TET_ROOT, on the other hand, is merely a logical structure in a directory under which test cases can be organized. Accordingly, test modules, as used in the claims, have a functional relationship with the harness as they are capable of being executed by the harness. TETware's directory element cannot be reasonably read as being executable or as having a functional relationship with the test case controller—it is merely a logical placeholder for test cases.

Therefore, among other things, and in connection with the other recited claim limitations, TETware fails to teach, suggest, or enable a "a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules, (ii) initiates execution of a connector to scan for and discover the plurality of available test cases that are stored in the one or more program modules and to organize the plurality of available test cases into a test case hierarchy, and (iii) receives user input indicating which of the plurality of available test cases in the test case hierarchy are selected test cases to be executed on the computer program," as now recited in some form in each of the independent claims. Because the "limited situation" identified by the Appeals Board no longer applies to the claims as recited, and because Applicants have further amended the claims to clearly distinguish the significant differences between TETware's directory element and Applicant's test module, Applicants respectfully submit that the 35 U.S.C. § 102(b) rejection over TETware has been overcome and should be withdrawn. Consequently, the rejections of record with respect to the dependent claims also have been overcome and similarly should be withdrawn. Also, as previously noted, *Hartmann* similarly fails to teach or suggest any of the recited features shown above to be lacking in the description of TETware. Accordingly, the 35 U.S.C. § 103(a) rejection should be withdrawn as well.

Based on at least the foregoing reasons, Applicants respectfully submit that the cited prior art fails to anticipate or make obvious Applicants invention, as claimed for example, in independent claims 1, 4, 15, and 24. Applicants note for the record that the remarks above render the remaining rejections of record for the independent and dependent claims moot, and thus addressing individual rejections or assertion with respect to the teachings of the cited art is unnecessary at the present time, but may be undertaken in the future if necessary or desirable, and Applicants reserve the right to do so.

In the event that the Examiner finds any remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney.

Dated this 28th day of October, 2008.

Respectfully submitted,

/GREGORY R. LUNT/

RICK D. NYDEGGER
Registration No. 26,651
ADRIAN J. LEE
Registration No. 42,785
GREGORY R. LUNT
Registration No. 57,354
Attorneys for Applicant
Customer No. 047973

AJL:GRL:cj
2131992_1.DOC